Automated Attribute Measurements of Buried Package Features in 3D X-ray Images using Deep Learning

Ramanpreet Singh Pahwa [1, 2], Ma Tin Lay Nwe [1], Richard Chang [1], Oo Zaw Min [1], Wang Jie [1], Saisubramaniam Gopalakrishnan [1], David Ho Soon Wee [3], Ren Qin [3], Vempati Srinivasa Rao [3], Haiwen Dai [4], Jens Timo Neumann [4], Ramani Pichumani [4], Tom Gregorich [4]
[1] Institute for Infocomm Research (I^2R), A*STAR
[2] Artificial Intelligence, Analytics And Informatics (AI^3), A*STAR
[4] Carl Zeiss SMT Inc.
[3] Institute of Microelectronics (IME), A*STAR
{ramanpreet_pahwa, tlnma, richard_chang, ozmin, wang_jie, g_saisubramaniam}@i2r.a-star.edu.sg,
{hosw, renq, vempati}@ime.a-star.edu.sg
{haiwen.dai, jens-timo.neumann, ramani.pichumani, tom.gregorich}@zeiss.com

***Abstract*—Deep Learning is being widely used to identify and segment various structures in 2D and 3D scans in fields such as robotics and medical imaging. We leverage this exciting technology to train state-of-the-art models for 3D object detection and segmentation for various buried structures such as Through Silicon Vias (TSVs), memory bumps, and logic bumps. We show in detail how we fabricate our wafers and generate 3D scans. Thereafter, we explain our approach in locating these different structures in 3D scans and how we further segment these structures into solders, voids, Cu-Pillars, and Cu-Pads for 3D metrology and defect identification. We compare our approach with state-of-the-art techniques and perform a thorough analysis to discuss the advantages and disadvantages of various approaches in each step.**

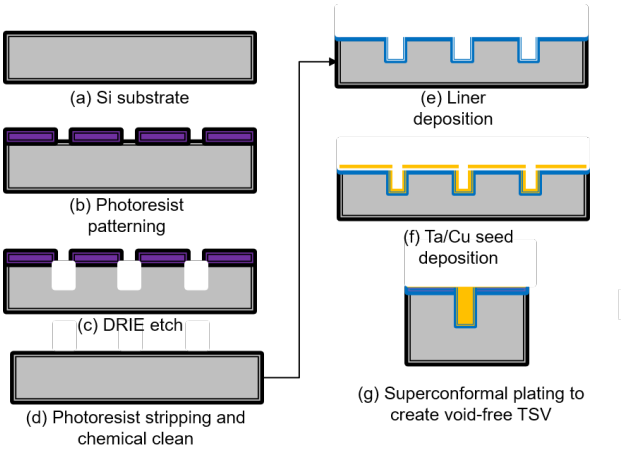
***Index Terms*—Deep Learning, 3D object detection, 3D segmentation, defect detection, 3D metrology**

I. Introduction

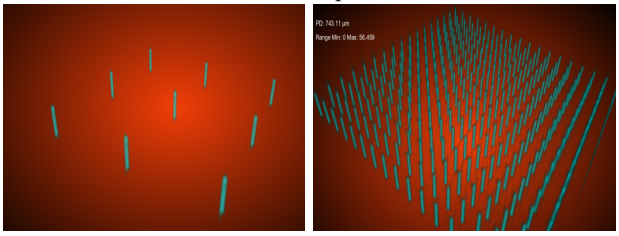
Failure analysis is a critical step in improving semiconductor manufacturing yields. With the advent of Industry 4.0 tools, there has been an increased focus on digital technology to improve yield across front-end and back-end manufacturers. With the increasing miniaturization of TSVs, RDLs, micro-bumps, and other package interconnects, detecting defects in these buried interconnects is simultaneously becoming more important and challenging. The most common approach currently for failure analysis is to cross-section semiconductor packages to identify internal process defects such as pad misalignment, unsolders, and solder shorts. However, cross-sectioning is a time-consuming, destructive, and difficult technique that provides information in a planar view only. The development of 3D X-ray microscopy (XRM) has provided the industry with the unique capability to image and investigate buried features such as micro-bumps, TSVs, and other metallic structures using a non-destructive technology that provides voxelized 3D-views. Machine and Deep learning have created significant impact on several technologies such as predictive maintenance, visual surveillance, object localization, and identification. Now semiconductor manufacturing is starting to see its influence [1], [2]. AI is fast becoming an integral part of advanced manufacturing and inspection fields such as component design [3], optical inspection [4], and anomaly detection [5]. The unique combination of 3D XRM and deep learning may soon establish a new gold-standard in package inspection and metrology.

In this paper, we will present our approach to automatically identify important interconnects in semiconductor packages and evaluate the characteristics of these interconnects. We will discuss our approach in fabricating defective and non-defective structures. Samples representative of stacked 2.5D packages are fabricated and assembled using thermo-compression bonding. Bonding parameters are varied to create packages with different solder fillet shapes, bond line thickness, and pad alignment scenarios. We follow up with our 3D measurement approach where a commercial 3D X-ray imaging tool is used to scan and generate high-quality tomographies of such packages. Thereafter, we focus on our AI based approach to automatically detect interconnects and measure characteristics directly in 3D. A two-step procedure is used for data analysis. We first identify the unique structures by estimating a 3D bounding box for each region of interest (TSV, Copper Pillar, Copper Pad, etc.). In the second step, we isolate these regions to perform multi-class 3D segmentation in these regions. The final step performs data cleaning and analysis to provide relevant 3D metrology information using these segmented regions. For instance, using robust 3D computer vision techniques we can measure the extent of solder extrusion and pad misalignment which are important characteristics for the chip fabrication and process control step. Based on the user's

This research is supported by Economic Development Board (EDB), Singapore under its IAF-ICP Grant no. I1901E0048 and administered by the Agency for Science, Technology and Research (A*STAR).

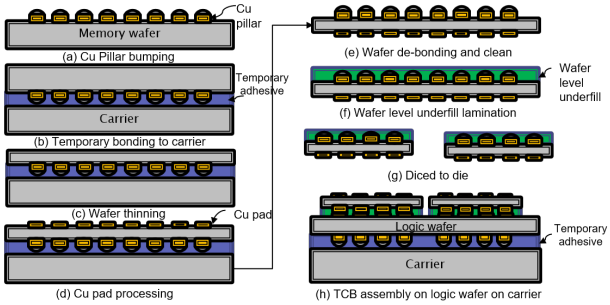


(a) Fabrication steps for TSVs.

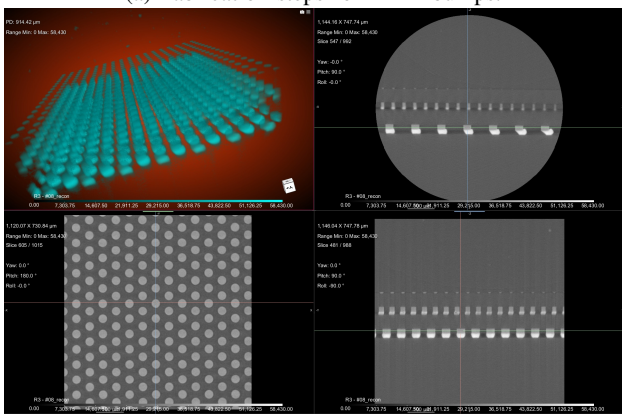


(b) Defective TSVs. (c) Normal TSVs.

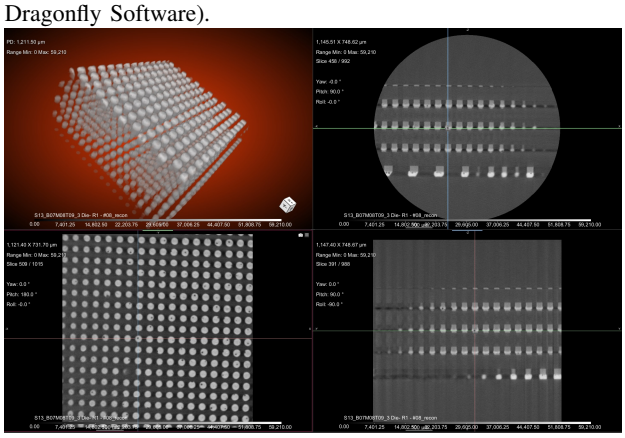
Fig. 1: Fabrication and 3D scans of Through Silicon Vias (TSVs).



(a) Fabrication steps for HBM bumps.



(b) Some examples of Memory and logic bumps (shown using Dragonfly Software).



(c) Some examples of Memory and logic bumps (shown using Dragonfly Software).

Fig. 2: Fabrication and 3D scans of HBM bumps.

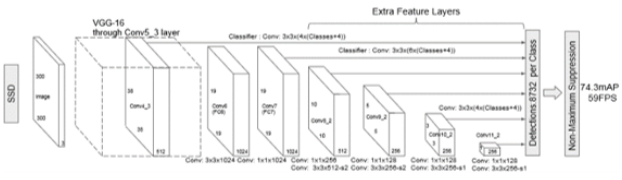
criteria, these structures can be potentially classified as either "pass" or "fail. As the focus of the paper is on the AI component of fault analysis, we will discuss the deep learning work in more details in the following sections.

II. Related Work

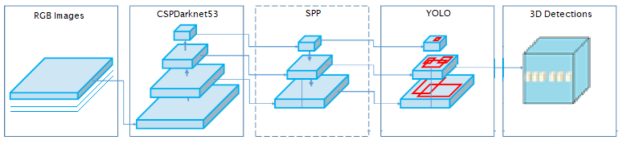
Many 2D object detectors have been introduced in the literature. Usual object detectors consist of two parts, a backbone which is pretrained on an existing dataset and an output network which is used to predict classes and bounding boxes. Output networks typically include two-stages networks such as Faster-RCNN [6] and its derivatives. One stage networks such as YOLO [7] and SSD [8] are also widely used. We use YOLOv4 [9] as it has demonstrated significantly higher performance compared to other networks and its previous versions. It consists of a backbone of CSPDarknet53 [10] with SPP [11] and YOLOv3 [12] as a dense predictor.

Recent literature on 3D segmentation, particularly in medical imaging, has focused primarily on using 3D U-Nets [13] or its variants, V-Net [14] and nnU-Net [15] [16]. DFKZ Net [16] employs an encoder to extract abstract representations and a decoder to recombine these representations. Myronenko et al. [17] uses an additional Variational Auto-encoder branch to regularize the shared encoder of U-Net. Cahall et al. [18] introduced Inception modules with learning objective of glioma sub-regions for medical imaging. Kamnitsas et al. considered ensembles of models and varying architectures and their aggregate predictions [19]. Isensee et al. argue that majority of proposed incremental improvements on bare-bone U-Nets do not seem to generalize over wide range of medical dataset, but tweak to a particular dataset. They propose a framework to analyze a given dataset to optimally set system parameters [15]. Milletari et al. [14] adopt volumetric convolution to process input data instead of slice-wise manner and also propose an adapted object function based on dice coefficient to counter annotation imbalance. Their work shows substantial improvement on both performance and convergence time.

We utilize a slice-and-fuse approach for 3D object detection.



(a) SSD architecture.



(b) Yolov4 architecture.

Fig. 3: 2D object detection modules used in our approach to identify 3D bounding boxes for each buried structure in 3D XRM scans.

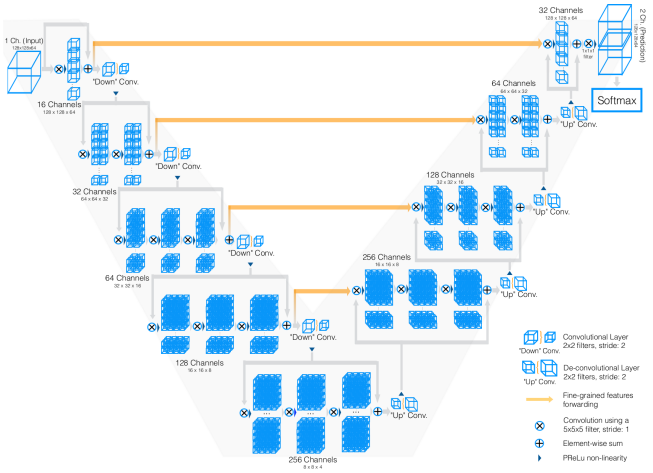


Fig. 4: Schematic representation of V-net architecture. [14]

We crop the region of interests and then perform dense multi-class 3D segmentation using nnU-Net, V-net, and our approach to compare the results.

III. Data Fabrication and Generation

In this section, we briefly describe how we fabricate and scan our 2.5D test vehicles. The test vehicles are designed to represent contemporary High-Performance Computing packages and utilize a silicon interposer with Through Silicon Vias (TSVs) and fine-pitch micro-pillar interconnects (logic and memory bumps). One of the major requirements for an AI based defect detection solution is to having access to enough defective data to identify and measure these defects such as solder voids.

Test vehicles consisting of TSVs are fabricated on $300mm$ silicon wafers using standard 2.5D manufacturing processes. For more details on TSV fabrication, the readers can refer to [2]. As shown in Fig. 1(a), we use a superconformal plating process to fill the Through Silicon Vias. Superconformal plating is used as this allows the base of the via to be filled at a superior rate that the sidewalls and the field. This results in a void-free via. Daisy-chain silicon chips are also produced to represent the DRAM and Logic dies which are assembled on top of each other to form High Bandwidth Memory (HBM) cubes. For more details on memory and logic bumps, the readers can refer to [2]. As shown in Fig. 2, both types of wafers are temporarily bonded to a carrier and thinned to $50m$. The memory wafers are eventually diced, and the chips are stacked individually on the logic wafer to form HBM cubes. We also utilize Thermo-compression bonding (TCB) to stack the memory dies on the logic wafer.

Once the chips are fabricated, we generate our 3D scans using 3D X-ray microscopy (XRM). 3D XRM provides us the capability to non-destructively inspect and scan the TSV and HBM cubes. The test vehicles are fitted on sample holders and positioned on the XRM autoloader. Thereafter, they are rotated incrementally from -3° through 183° . Raw 2D X-ray scans are imaged. These 2D scans along with geometrical information is used in a propriety algorithm to computationally obtain 3D X-Ray scans. This process, also known as computational tomography, allows the datasets to be visualized in 3D where buried structures can be imaged at a high resolution. For more details on data generation, the readers can refer to [1], [2]. The data output is approximated $1,000 \times 1,000 \times 1,000$ voxels.

IV. 3D object detection

Our first module's objective is to extract the 3D structures in full resolution XRM scans. Hypothetically, we could segment the full resolution scans directly. However, this is computationally not possible currently with the available hardware. Running a traditional 3D convolutional neural networks (CNN) on a scan consisting of $512 \times 512 \times 512$ voxels runs out of memory while training a deep learning model. Hence, our approach first identifies regions of interests that contain structures as TSVs, logic and memory bumps. This is done by returning a 3D bounding box per identified structure.

We approach our problem using a slice-and-fuse approach to significantly reduce the required memory load. As processing 3D scans is extremely memory intensive, we leverage on mature 2D object detection techniques along with geometrical consistency across neighborhood 2D scans. First, we slice the 3D scans into 2D images. We use a custom-trained 2D object detector, YOLOv4, to detect buried structures in 2D images. Thereafter, we generate 3D bounding boxes from these 2D detections utilizing neighborhood information. We use YOLOv4 [9] and SSD [8] as our two model architectures.

YOLOv4 consists of a backbone of CSPDarknet53 [10] with SPP [11] and YOLOv3 as a dense predictor. The object detector first outputs 2D bounding boxes defined by [x, y, w, h]^i. Here [x, y] represents the mid point of the bounding box, [w, h] represents the width and height of the bounding box and i represents i^{th} bounding box in the 2D scan. We concatenate the 2D results along z-direction to obtain 3D bounding boxes. Fig. 3 shows the architecture of the network and the object detection pipeline. We use a pre-defined threshold to concatenate 2D bounding boxes into a

consolidated 3D bounding box. We also leverage on known information that the structures are fabricated and packaged at about similar distances from each other and consist of similar size. This information is used as part of structure recovery (SR) module to further refine our 3D bounding boxes.

V. 3D SEGMENTATIONS

In this section, we describe our approach to perform individual segmentation of different components of TSVs, Logic die, and Memory die. This can be done in various ways. We can slice the voxels into 2D planes and treat each plane as an individual image. We can use multi-class segmentation on each image and combine the results as part of post processing step to create 3D segmentations. However, such an approach does not use geometrical consistency across neighbouring voxels and hence, may perform poorly for sophisticated structures such as memory die. One can also perform a series of binary segmentations such as identifying voxels belonging to logic die. This can be followed by identifying Cu-Pillar, solder in two parallel steps and finally identifying voids that may be present in solder area. We partially adopt this strategy in our Ensemble of 2-step Cascaded Neural Network as will be described below. We implement two state-of-the-art 3D multi-class segmentation approaches namely V-net [14] and nnU-Net [15]. We hope that this work provide the community with a comprehensive analysis on strengths and weaknesses of each network on semiconductor datasets.

A. V-net

V-net uses volumetric convolution aiming to extract features with respect to 3D topology of input data. Although simple in its architecture, it is considered a baseline structure and sets a groundwork for our experiment suite. Similar to U-net, the model forwards feature compressed in various scale to its decompression half (Fig. 4). The model captures both coarse and fine latent information for better contour prediction.

With its novel objective function adapted from dice coefficient, the model mitigates the negative effect brought by imbalanced annotation. This process is carried out without manual assignment of class weights on background and foreground materials. We observe experiments with concrete improvement from ones using the same network trained optimizing re-weighted cross-entropy loss. The input volumes are normalized to a fixed size of $64 \times 64 \times 64$ voxels. The final network consists of 8.2 million parameters. Under such setup, we carry out both non-augmented and augmented experiments on TSV, memory, and logic die multi-class segmentations.

B. nnU-Net

We also use nnU-Net [15] to segment TSV, memory, and logic die RoIs produced by the object detection module. The network analyzes key properties of the dataset such as image size, voxel spacing information etc. to automatically fine-tune fundamental network parameters such as loss function and training schedule. The network also adapts its topology, patch size, batch size and other important parameters based

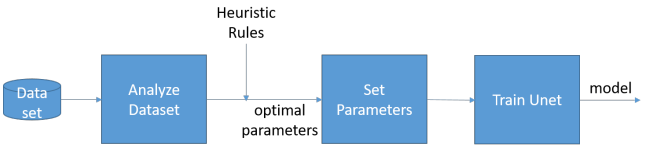


Fig. 5: Illustration of nnU-Net framework.

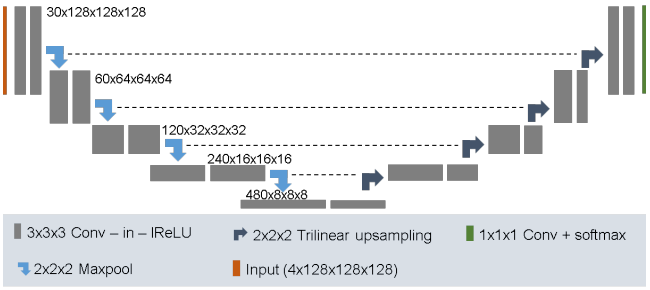


Fig. 6: Illustration of bare-bone U-Net [15]

on the GPU memory of the underlying system to perform 3D segmentation robustly. Hence, we do not need to heuristically set system parameters once dataset changes. We let the network choose the best parameters and only compare our results directly with other approaches. Firstly, we check the integrity of the dataset to identify and eliminate any outliers that may exist in the dataset. Then, we use nnU-Net module to train a 3D segmentation model for multi-class segmentation on our datasets. Data augmentation is also performed on-the-fly by nnU-Net. Basic data augmentation methods such as scaling, mirroring, rotating, gamma correcting and elastic deforming are used. We turn on and off this setting to generate results with and without augmentation. After autonomously setting the network hyper-parameters, the optimally designed 3D U-Net performs training by itself. We choose a single 3D U-Net architecture instead of a cascade approach. This results in the final network consisting of 16.06M trainable parameters. The network generally performs five-fold cross-validation. However, we manually set validation data to compare performances of the three networks objectively.

C. Ensemble of 2-step Cascaded AnIsotropic CNN (En-caiCNN)

Here, we extended our method based on Cascaded AnIsotropic Convolutional Neural Network (caiCNN) [20] for 3D voxel segmentation. The advantage of this method is the possibility to limit the focus area to detect a particular RoI. As illustrated in Fig. 7, Cu-Pillar is the subset of whole component for logic or memory die. Thus, we can force the network to look only inside foreground region to detect Cu-pillar. Using logic die as an illustration, we present an Ensemble of 2-step caiCNN (referred to as En-caiCNN) in Fig. 7 for 3D voxel segmentation. In the first step, foreground region (entire test vehicle) is segmented from the background. In the second step, each of the three regions is detected inside the foreground object by corresponding binary nets, C_e-Net, S_e-Net, and V_e-

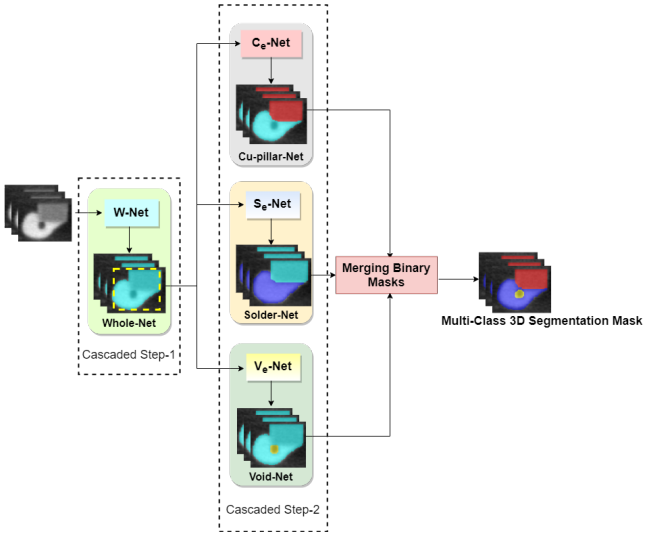


Fig. 7: Our Ensemble of 2-step Cascaded AnIsotropic Convolutional Neural Network (En-caiCNN).

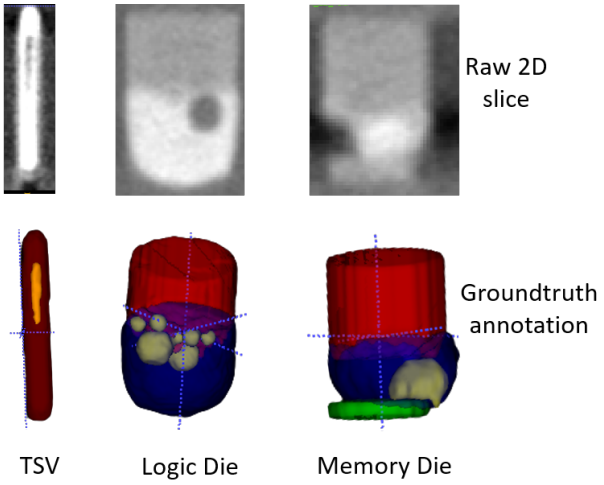


Fig. 8: A sample of our manual data annotation for TSV, Logic die, and Memory die.

Net. These three binary nets are trained on the same the RoI as shown in Fig. 7. While C_e-Net learns Cu-Pillar as its target class, S_e-Net and V_e-Net learn solder and void regions as their target classes respectively. All three binary nets consider the remaining voxels (that are not classified as foreground) as background. Finally, all the binary masks are merged to produce the final multi-class segmentation mask. The Ensemble method allows the network to focus only on one type of region inside the foreground object. To detect the voxels in TSV, we remove C_e-Net and S_e-Net while for memory, we add one more neural network, P_e-Net to detect the additional component - CuPad.

VI. Experiments

In this section, we report our results for two major steps – 3D Object detection and 3D multi-class segmentation. As shown in as shown in Table I, II, we divided our data into 3 groups: training data, validation data, and testing data for our experiments. We report our results on 3 object detection techniques and 3 3D segmentation techniques. In both categories, we adopt and modify the state-of-the-art techniques for further improvement by identifying the gaps in these approaches.

TABLE I: The division of annotated data into train, validation, and test subset for our experiments for 3D object detection.

Components	Train	Val	Test
TSV	221	0	56
Logic Die	1014	0	169
Memory Die	1734	0	289

A. Data annotation

As there are no mature open-source tools for labelling data in 3D quickly and accurately, we decided to use a combination of internal tools and open-source tools for data labelling, visualization, and quality assessment. Along with some assistance of labelling tool, we manually labelled the 3D bounding boxes for all different components of each chip – namely – TSVs, Logic die, and memory die. This step is quite fast and mechanically easy to perform. Hence, we labelled approximately $275, 1200, 2000$ samples for TSV, Logic die, and memory die respectively. The small number of annotations for TSVs are due to having limited fabrication of TSVs. Table I displays our annotation information for 3D object detection. For object detection, we decided not to use a validation dataset as the SSD module does not use it. Thus, the entire data is divided into train and test data. This results in a fair comparison between SSD an YOLO object detection modules.

TABLE II: The division of annotated data into train, validation, and test subset for our experiments for 3D multi-class segmentation.

Components	Train	Val	Test
TSV	5	2	6
Logic Die	8	4	18
Memory Die	8	4	18

Once we obtain the 3D bounding box regions, these 3D regions were cropped and extracted. Each voxel present inside these cropped regions is assigned a label – background, TSV, Copper-Pillar (CuP), Copper-Pad (CuPad), solder, or void. The voids are present inside the solder region and may touch the neighboring surface of CuP or CuPad. However, the voids cannot be outside the solder region. Our labelled dataset includes TSVs, logic and memory dies with and without voids. Table II displays our annotation information for 3D segmentation. Our annotation strategy ensures that we selected approximately 50% void and 50% non-void test vehicles for labelling. Fig. 8 shows an example of the 3D annotations for TSV, logic bump and memory bump with voids.

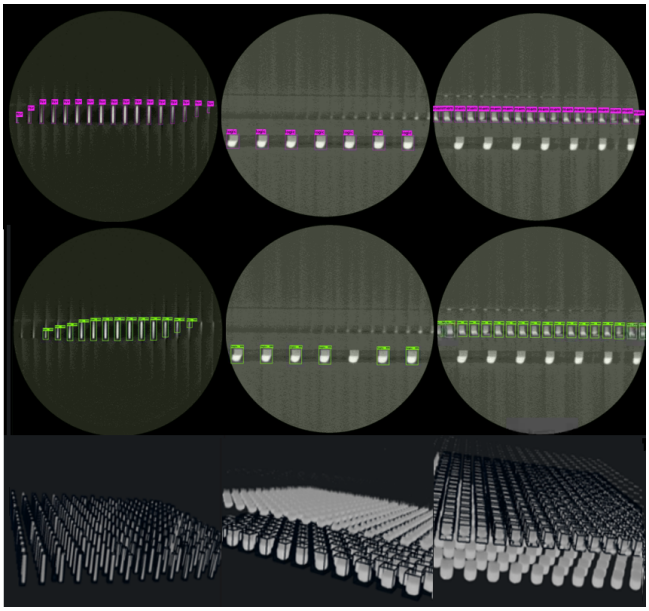


Fig. 9: 2D and 3D results for each structure TSV (left), logic die (center) and memory die (right). The first two rows show the 2D results from SSD and YOLOv4 respectively. 3D bounding boxes have been displayed on the structures on the last row.

For our 3D segmentation experiments, initially, we had uniformly mixed void and non-void test vehicles in train, validation, and test datasets. However we noticed an abnormally high dice loss in our test results. Upon further investigation, we observed that including non-void test vehicles in training data was the root-cause of this behavior. It resulted in various erroneous segmentations in validation and test structures. As this was observed in both memory and logic die experiments, we decided to remove non-void data from training dataset. This resulted in slightly lower training data accuracy but significantly better accuracy in validation and test datasets. In our experiments, we include two samples with voids in validation dataset for logic and memory die. We include only 1 sample with void for validation data for TSVs due to small amount of data availability for TSVs. The remaining set of samples that may or may not contain voids are used as test dataset. One possible reason for such behavior could be the small number of labelled samples available to us for 3D segmentation experiments. Moreover, the dice score is heavily penalized when we detect voids where none are present. This could further aggravate the issue in original data distribution related experiments.

B. 3D object detection

In this subsection, we will present the results from the 3D object detection module. In total, we annotated 277 TSVs, $1,183$ logic die and $2,023$ memory die to test this module. We split the data into $80\% - 20\%$ for training and testing sets respectively. The models are trained for maximum 6000 iterations. The momentum and weight decay are set as 0.9 and 0.000 respectively. The step decay learning rate is adopted with initial learning rate 0.01 and multiplied by a factor 0.1 at steps $4,800$ and $5,400$. The backbone network's pre-trained weights on MS-COCO [21] are updated using our training data. As described in the previous section, the structures are first sliced into around $1,000$ images each. Then, 2D detection is performed on the images, and the 2D results are processed to generate 3D bounding boxes on the whole scan. To demonstrate the accuracy improvement from our structure recovery (SR), we compare our approach along with SR with SSD and YOLOv4 methods. All the three detectors are evaluated using an Intersection-Over-Union (IoU) metric and mean Average Precision (mAP). We use an IoU threshold of 0.5 to determine whether the prediction is correct or not. A correct prediction is categorized as True Positive and an incorrect prediction is categorized as False Positive. The precision of our detector is defined as:

$$P = \frac{TP}{TP + FP} \tag{1}$$

Here, P refers to Precision, TP and FP refer to True and False positives respectively. Our results show that all structures are detected accurately and extracted properly by our approach. Table III shows the results of the object detection module on TSVs, logic, and memory die. Fig. 9 shows the overall 2D detections results. We observe that YOLO performs better on slices where TSV structures are smaller and also results in less mis-detections compared to SSD. The 3D detection results are shown in the last row. Our method outperforms both YOLO and SSD in 3D thanks to our structure recovery approach.

TABLE III: Results for our 3D object detection approaches on our three test vehicles.

Method	TSV		Logic Die		Memory Die	
	2D mAP	3D IoU	2D mAP	3D IoU	2D mAP	3D IoU
SSD	0.955	0.697	0.999	0.686	0.986	0.610
YOLOv4	0.998	0.952	0.997	0.942	0.993	0.903
YOLOv4-SR	**0.998**	**0.952**	**0.997**	**0.963**	**0.993**	**0.935**

C. V-Net

We perform experiments using V-Net architecture as discussed in the previous Section. The end-to-end training process aims to categorize pixels into copper pillar, copper pad, solder, TSV and void regions. Dice metric is used to evaluate the performance on the test vehicles. The results are presented in Table IV, V, VI. The input volumes are normalized to a fixed size of $64 \times 64 \times 64$ voxels. The final network consists of 8.2 million parameters. Under such setup, we carry out both non-augmented and augmented experiments on TSV, memory, and logic die multi-class segmentations. The focus of the evaluation lies on the comparison of convergence rates, respective qualitative inferences, and eventual performance. We

TABLE IV: Results for our 3D multi-class segmentation for TSVs.

w/o Aug	Solder			Void		
	Train	Val	Test	Train	Val	Test
V-net	0.942	**0.915**	0.907	0.899	**0.883**	**0.886**
nnU-Net	**0.994**	0.908	0.918	**0.985**	0.861	0.836
En-caiCNN	0.971	0.914	**0.928**	0.834	0.844	0.814
with Aug	Solder			Void		
V-net	0.957	**0.921**	**0.924**	0.919	**0.898**	**0.906**
nnU-Net	**0.994**	0.913	0.923	**0.986**	0.860	0.770
En-caiCNN	0.870	0.903	0.921	0.792	0.820	0.651

TABLE V: Results for our 3D multi-class segmentation for Logic Die.

w/o Aug	Cu-Pillar			Solder			Void		
	Train	Val	Test	Train	Val	Test	Train	Val	Test
V-net	0.974	0.908	0.912	0.947	0.901	0.895	0.854	0.786	0.805
nnU-Net	**0.986**	**0.925**	**0.933**	**0.979**	**0.938**	**0.931**	**0.943**	**0.883**	**0.855**
En-caiCNN	0.958	0.910	0.924	0.941	0.910	0.925	0.914	0.857	0.796
With Aug	Cu-Pillar			Solder			Void		
V-net	0.982	0.911	0.923	0.979	0.896	0.902	0.848	0.826	0.833
nnU-Net	**0.987**	**0.924**	**0.933**	**0.980**	**0.942**	**0.932**	**0.947**	**0.888**	**0.857**
En-caiCNN	0.948	0.898	0.903	0.952	0.927	0.919	0.898	0.875	0.824

TABLE VI: Results for our 3D multi-class segmentation for Memory Die.

w/o Aug	Cu-Pillar			Cu-Pad			Solder			Void		
	Train	Val	Test	Train	Val	Test	Train	Val	Test	Train	Val	Test
V-net	0.927	**0.861**	**0.878**	0.826	0.622	**0.631**	0.874	**0.815**	0.809	0.832	**0.804**	0.789
nnU-Net	**0.985**	0.837	0.866	**0.982**	**0.640**	0.625	**0.967**	0.786	0.803	**0.953**	0.772	**0.817**
En-caiCNN	0.929	0.852	0.825	0.827	0.535	0.458	0.880	0.799	**0.816**	0.855	0.797	0.775
With Aug	Cu-Pillar			Cu-Pad			Solder			Void		
V-net	0.942	**0.883**	**0.898**	0.851	0.623	0.619	0.866	**0.807**	**0.811**	0.822	**0.783**	0.795
nnU-Net	**0.985**	0.831	0.857	**0.982**	**0.636**	**0.621**	**0.966**	0.779	0.799	**0.952**	0.757	**0.823**
En-caiCNN	0.927	0.870	0.853	0.840	0.519	0.455	0.884	0.773	0.782	0.825	0.743	0.755

uses random mirroring, rotation, and scaling for augmenting our data.

Our XRM dataset also shows a signature characteristic of containing sparse and low-density RoI. We observe relative worse result on voids even when dealing with cropped data where the region of interest is considered much larger compared to original data. We observe experiments with concrete improvement from ones using the same network trained optimizing re-weighted cross-entropy loss. We observe that by applying some geometric augmentations, the overall dice evaluation result has slight improvement compared to ones without augmentation. Although the effect of augmentation is very dependent on the quality and quantity of the original dataset, we notice irregular-shape regions such as solder and void tend to bring in more noise when we apply augmentation.

D. nnU-Net

We also perform multi-label segmentation experiments by nnU-Net. As we chose single U-Net architecture, we only perform end-to-end training of one U-Net to detect individual components such as copper pillar, copper pad, solder, TSV, and voids. The results are presented in Table IV, V, VI using Dice score as our metric of choice. We observe that nnU-Net, in general, gives the best and most consistent performance among all three networks. This is due to the network fine-tuning the hyper parameters to select the best learning strategy for our dataset. nnU-Net only gives slightly inferior performance in segmentation of Copper Pillars for memory die and for solder regions for Logic die.

E. En-caiCNN

Similarly, we conduct experiments using our custom En-caiCNN model architecture. We first detect whole component in each of TSV, logic, and memory die. In the second step, we detect individual regions: copper pillar, copper pad, solder, and voids inside the whole component individually as shown in Fig. 7. Each 2-step cascaded CNN outputs a mask for each region inside the whole component. We fuse the masks for 3 regions (for logic die) to generate the voxel segmentation mask for all 3 classes. The 3D segmentation dice score for Ensemble method is listed in Table IV, V, VI. Our custom network provides the best results for TSV segmentation when training without augmentation. It also provides the best test results for Solder segmentation in Logic Die. The results degrade significantly when augmentation is added to the network for certain components such as TSV voids. Overall, augmentation seems to have a negative impact on dice scores for our network. This is contrary to what is generally expected as we can see from improvement in results with augmentations for V-Net and nnU-Net. In future, we intend to investigate these findings in order to further improve our segmentation results.

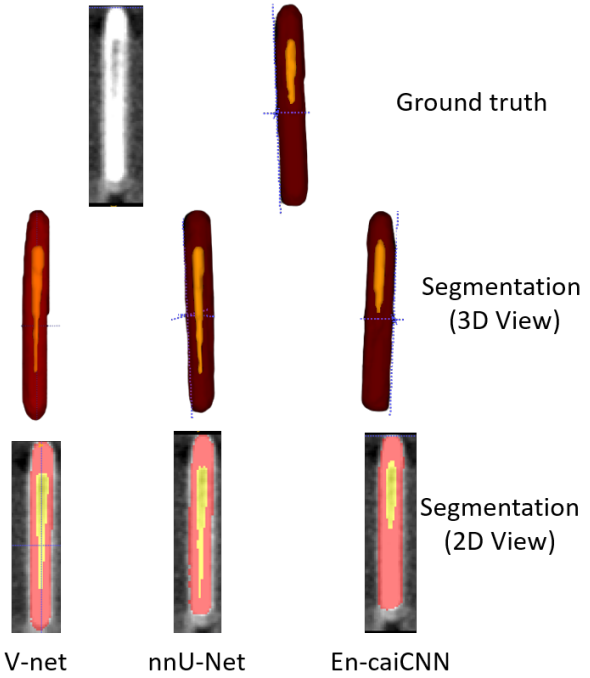


Fig. 10: 3D segmentation results using our three different networks for TSV. The first row displays a sample raw image and groundtruth annotation. Second row shows our inferred 3D segmentation using three different networks and the final rows shows the corresponding 2D segmented slice for visual comparison

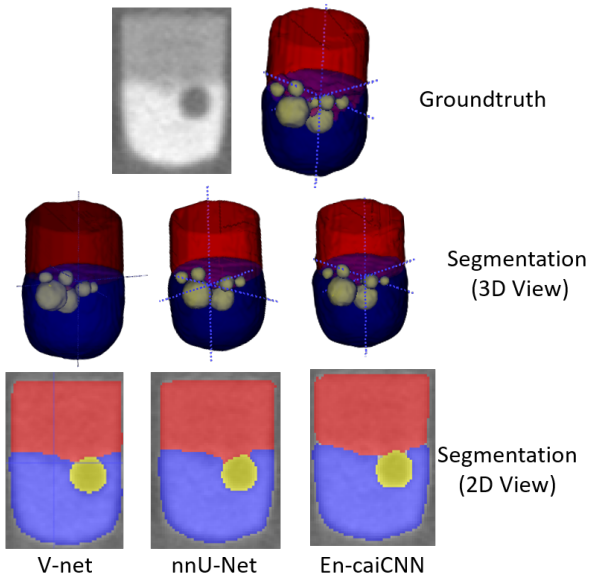


Fig. 11: 3D segmentation results using our three different networks for Logic Die. The first row displays a sample raw image and groundtruth annotation. Second row shows our inferred 3D segmentation using three different networks and the final rows shows the corresponding 2D segmented slice for visual comparison.

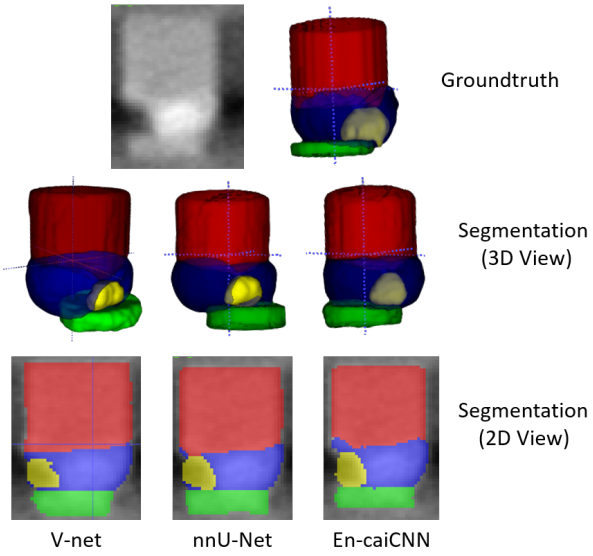


Fig. 12: 3D segmentation results using our three different networks for Memory Die. The first row displays a sample raw image and groundtruth annotation. Second row shows our inferred 3D segmentation using three different networks and the final rows shows the corresponding 2D segmented slice for visual comparison

VII. Conclusion

In this paper, we have presented a novel deep learning based defect detection approach for buried structures in 3D X-ray scans. We presented our strategy of fabricating and scanning TSVs, logic and memory bumps. Thereafter, we explained in detail how we perform 3D object detection followed by 3D segmentation and 3D metrology to provide useful information for these three structures. Our framework that includes 3D deep learning based object detection, segmentation and metrology on high resolution 3D voxelized XRM scans serves as a baseline starting point for future multi-class based 3D metrology. Our approach is capable of perform defect detection on high resolution 3D X-ray scans that is not possible with conventional 3D CNNs. In future, we intend to add more annotated samples along with extensive 3D data augmentation strategy to further improve the void segmentation results and detect other types of defects such as pad misalignment and solder extrusions. Semi Supervised learning and Few shot learning can also assist in improving the dice scores further in future.

Acknowledgment

We would like to thank IME, A*STAR and Carl Zeiss SMT Inc. for their invaluable support and expertise in designing the fabrication and producing the 3D scans. This work is supported by Economic Development Board (EDB), Singapore the under its IAF-ICP Grant no. I1901E0048 and administered by the Agency for Science, Technology and Research.

REFERENCES

[1] R. S. Pahwa, T. L. Nwe, R. Chang, W. Jie, O. Z. Min, S. W. Ho, R. Qin, V. S. Rao, Y. Yang, J. T. Neumann, R. Pichumani, and T. Gregorich, "Deep learning analysis of 3d x-ray images for automated object detection and attribute measurement of buried package features," in *IEEE 22nd Electronics Packaging Technology Conference (EPTC)*, 2020, pp. 221–227.

[2] R. S. Pahwa, T. L. Nwe, R. Chang, W. Jie, O. Z. Min, S. W. Ho, V. S. Rao, Y. Yang, J. T. Neumann, R. Pichumani, and T. Gregorich, "Machine-based learning methodologies for 3d x-ray measurement, characterization and optimization for buried structures in advanced ic packages," in *17th International Wafer Level Packaging Conference (IWLPC)*, 2020.

[3] J. Wang, Y. Ma, L. Zhang, R. X. Gao, and D. Wu, "Deep learning for smart manufacturing: Methods and applications," *Journal of Manufacturing Systems*, vol. 48, pp. 144 – 156, 2018, Special Issue on Smart Manufacturing.

[4] D. Weimer, B. Scholz-Reiter, and M. Shpitalni, "Design of deep convolutional neural network architectures for automated feature extraction in industrial inspection," *CIRP Annals - Manufacturing Technology*, vol. 65, 05 2016.

[5] R. S. Pahwa, J. Chao, J. Paul, Y. Li, M. T. Lay Nwe, S. Xie, A. James, A. Ambikapathi, Z. Zeng, and V. R. Chandrasekhar, "Faultnet: Faulty rail-valves detection using deep learning and computer vision," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, pp. 559–566.

[6] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds., pp. 91–99. Curran Associates, Inc., 2015.

[7] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *CoRR*, vol. abs/1804.02767, 2018.

[8] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C.-Y. Fu, and A. C. Berg, "SSD: single shot multibox detector," *CoRR*, vol. abs/1512.02325, 2015.

[9] A. Bochkovskiy, C-Y Wang, and H.Y. Liao, "Yolov4: Optimal speed and accuracy of object detection," 04 2020.

[10] C-Y. Wang, H-Y M. Liao, I-H. Yeh, Y-H. Wu, P-Y. Chen, and J-W. Hsieh, "Cspnet: A new backbone that can enhance learning capability of cnn," 2019.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *CoRR*, vol. abs/1406.4729, 2014.

[12] A. Lemay, "Kidney recognition in ct using yolov3," 2019.

[13] Özgün Çiçek, Ahmed Abdulkadir, Soeren S. Lienkamp, Thomas Brox, and Olaf Ronneberger, "3d u-net: Learning dense volumetric segmentation from sparse annotation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016*, Sebastien Ourselin, Leo Joskowicz, Mert R. Sabuncu, Gozde Unal, and William Wells, Eds., Cham, 2016, pp. 424–432, Springer International Publishing.

[14] F. Milletari, N. Navab, and S. Ahmadi, "V-net: Fully convolutional neural networks for volumetric medical image segmentation," in *2016 Fourth International Conference on 3D Vision (3DV)*, 2016, pp. 565–571.

[15] Fabian Isensee, Jens Petersen, André Klein, David Zimmerer, Paul F. Jaeger, Simon Kohl, Jakob Wasserthal, Gregor Koehler, Tobias Norajitra, Sebastian J. Wirkert, and Klaus H. Maier-Hein, "nnu-net: Self-adapting framework for u-net-based medical image segmentation," *CoRR*, vol. abs/1809.10486, 2018.

[16] Fabian Isensee, Philipp Kickingereder, Wolfgang Wick, Martin Bendszus, and Klaus H. Maier-Hein, "Brain tumor segmentation and radiomics survival prediction: Contribution to the brats 2017 challenge," in *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*, Alessandro Crimi, Spyridon Bakas, Hugo Kuijf, Bjoern Menze, and Mauricio Reyes, Eds., Cham, 2018, pp. 287–297, Springer International Publishing.

[17] Andriy Myronenko, "3d mri brain tumor segmentation using autoencoder regularization," in *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*, Alessandro Crimi, Spyridon Bakas, Hugo Kuijf, Farahani Keyvan, Mauricio Reyes, and Theo van Walsum, Eds., Cham, 2019, pp. 311–320, Springer International Publishing.

[18] Daniel E Cahall, Ghulam Rasool, Nidhal C Bouaynaya, and Hassan M Fathallah-Shaykh, "Inception modules enhance brain tumor segmentation," *Frontiers in Computational Neuroscience*, vol. 13, pp. 44, 2019.

[19] K. Kamnitsas, W. Bai, E. Ferrante, S. McDonagh, M. Sinclair, N. Pawlowski, M. Rajchl, M. Lee, B. Kainz, D. Rueckert, and B. Glocker, "Ensembles of multiple models and architectures for robust brain tumour segmentation," in *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*, Alessandro Crimi, Spyridon Bakas, Hugo Kuijf, Bjoern Menze, and Mauricio Reyes, Eds., Cham, 2018, pp. 450–462, Springer International Publishing.

[20] T. L. Nwe, O. Z. Min, R. Chang, D. Lin, S. Prasad, S. Dong, Y. Li, and R. S. Pahwa, "On the use of component structural characteristics for voxel segmentation in semicon 3D images," in *under review in IEEE International Conference on Image Processing (ICIP)*, 2021.

[21] T.-Y. Lin, M. Maire, S.J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. Lawrence Zitnick, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014.